Ann Math Artif Intell
DOI 10.1007/s10472-009-9122-0

Classifying the multi robot path finding problem into a quadratic competitive complexity class

Shahar Sarid · Amir Shapiro

Abstract We explore two motion planning problems where a group of mobile robots has to reach a target located in an a priori unknown environment while on-line planning the next step. In the first problem the target position is unknown and should be found by the robots, while in the second problem the target position is known and only a path to it should be found. We focus on optimizing the cost of the task in terms of motion time, which, under the assumption of uniform velocity of all the robots, correlates to the path length passed by the robot which reaches the target. The performance of an on-line algorithm is usually expressed in terms of *Competitiveness*, the constant ratio between the on-line and the optimal off-line solutions. Specifically, the ratio between the lengths of the actual path made by the robot which reached the target to the shortest path to the target. We use generalized competitiveness, i.e., the ratio is not necessarily constant, but could be any function. Classification of a motion planning task in the sense of performance is done by finding an upper and a lower bounds on the competitiveness of all algorithms solving that task. If the two bounds belong to the same functional class this is the *Competitive Complexity Class* of the task. We find the two bounds for the aforementioned common on-line motion planning problems, and classify them into competitive classes. It is shown that in general any on-line motion planning algorithm that tries to solve these problems must have at least a quadratic competitive performance. This is a lower bound of

S. Sarid (✉) · A. Shapiro
Department of Mechanical Engineering,
Ben Gurion University of the Negev, Beer-Sheva, Israel
e-mail: sarids@bgu.ac.il

A. Shapiro
Paul Ivanier Center for Robotics Research and Production,
Perlstone Center for Aeronautical Engineering,
Ben Gurion University of the Negev, Beer-Sheva, Israel
e-mail: ashapiro@bgu.ac.il

Published online: 11 March 2009

the problems. This paper describes two new on-line navigation algorithm which solve the problems under discussion. The first is called *MRSAM*, short for *Multi-Robot Search Area Multiplication*, and the second is called *MRBUG*, short for *Multi-Robot BUG* which extends Lumelsky famous BUG algorithm. Both algorithms have quadratic upper bounds, which prove that the problems they solve have quadratic upper bounds. Thus it is shown that navigation in an unknown environment by a group of robots belongs to a quadratic competitive class. *MRSAM* and *MRBUG* have a quadratic competitive performance and thus have optimal competitiveness. The algorithms' performance is simulated in office-like environments.

Keywords Multi-robot · Motion planning · Competitive complexity class · Optimal algorithm · Lower bound · On-line

Mathematics Subject Classifications (2000) 68Q25 · 68Q15 · 68Q17 · 68W15 · 68W40 · 68T40 · 11Y16 · 93C85

1 Introduction

The problems of finding or reaching a target in an unknown planar environment are very important in many practical and academic research fields. The most significant domains are humanitarian robotics, industry robotics and military robotics where demining [25] is a good example for the last two. Area coverage is a corresponding task, since the searching unit will cover a certain area before finding the target,[1] and when the number of targets in a specific area is unknown, the whole area must be covered. Examples for typical applications are demining, search and rescue missions, cleaning supermarkets and train stations, detecting contaminated or radioactive substances in factories, nuclear reactors, or in the open field, planetary exploration and sample acquisition. This paper is concerned with the aforementioned problems solved by multiple mobile robots.

In multi mobile robot motion tasks the physical travel time is much more significant than on-board computation time. Assuming the robot's velocity stays constant at all times, travel time correlates to path length. Hence, under the assumption of uniform velocity among all robots, travel time corresponds to the path length of the robot that found the target. Since all the robots travel the same path length per time unit, travel time corresponds to the path length of any robot that terminated at the same time the target was found. We denote the distance traveled by each robot, l, and the optimal off-line solution l_{opt}. Under a uniform power output assumption, travel time corresponds to path traversibility cost. Hence the algorithms discussed in this paper are being analyzed in terms of length or cost of the path traveled by one robot during algorithm execution.

In the problems discussed above, using a group of robots can have many advantages over using only one robot. The most important advantage is the shortening of

[1] When dealing with limited sensors which detects the target upon arrival or within a specific constant range from the target which is much smaller in magnitude than the typical length of the problem's environment.

the total mission time, another advantage is increased robustness, since the multitude of robots can easily overcome a malfunction in one or more of the units. The decrease of the individual mechanical wear and power consumption per mission maximizes the life span of each robot and prolongs the whole mission duration and range. Other advantages concerns maintaining radio connectivity between the robots and the base station [1]. Another advantage of a group of robots is a decreased sensor uncertainty due to merging of overlapping information, it has been shown in [6] that multiple robots localize themselves more efficiently, especially when they have different sensor capabilities.

The purpose of this paper is to introduce and to prove optimality of two new algorithms using multiple robots in an unknown planar environment. The first is *MRSAM*, which solves the problem of finding a target whose position is unknown, and the second is *MRBUG*, which solves the problem of finding a path to a target whose position is known. The notion of competitiveness compares the performance of an on-line algorithm to the optimal off-line solution for the same problem. In particular, an algorithm for a task P is said to be competitive if its solution to every instance of P is bounded by a constant times l_{opt} [17]. Generalized *Competitive Complexity* and *Competitive Complexity Classes* are introduced and discussed in [10]; however, most research dealing with competitiveness strive to identify specific classes of environments in which constant competitiveness can be achieved. In contrast, our objective is to identify the competitive relationship governing the fully general on-line navigation problem for multiple robots. Optimality is proved by showing that the problems themselves belong to the quadratic competitive complexity class and that the performance of *MRSAM* and *MRBUG* belongs to that class too. *MRSAM* is based on the area doubling strategy of *SAD1* algorithm introduced in [10], which launches one robot to search for a target whose position is unknown in an unknown environment. *SAD1* assigns a search disc to the robot, which is doubled at each step, if the target is not found. *MRBUG* is based on growing bounding ellipses used in the *CBUG* algorithm for one robot [11] to restrict the search for the path to a target whose position is known in an unknown environment.

Recent work related to the subject of mobile multi-robot motion planning deals with many aspects of the problem. A major issue is whether the group architecture is centralized [29, 30] or decentralized [5, 23], i.e., whether there is only one control agent, or not. In the second case each robot is autonomous and there is neither a centralized component, nor any other global coordination needed. In distributed systems [14, 28], robots work concurrently on solving the tasks, sometimes helping each other [27]. Communication is a very close subject [4, 19], since, when there is no communication between the robots or when it is limited [2, 7], the system cannot be centralized. Intermediate systems represent real-world setups better, for example, the semi-decentralized approach in [21], where robot teams cover the space independent of each other, but robots within a team communicate state and share information. Limited communication plays an important role when dealing with ant-like robots, where messages between the robots are passed mainly or only through marking they leave on the terrain, [18].

A solution to a problem can change according to the availability of information on the environment prior to algorithm execution. Online solutions assume no knowledge of the environment when the algorithm starts, while off-line solutions rely on a priori knowledge. An off-line algorithm is presented in the notable early paper

[20], and a new work, [15], in that area focuses on robustness, and completeness of the algorithm. Robustness measures the performance in case of failures [19] and an algorithm is considered complete if for any input it correctly reports whether or not there is a solution in a finite amount of time [12, 13]. A limited-communication complete algorithm is presented in [24]. The solution presented in this paper is complete and robust and can be decentralized or centralized, depending on the setup communication.[2] *MRSAM* algorithm appeared first in [26], and here its detailed performance analysis and the simulation results are introduced for the first time.

The structure and contributions of the paper are as follows. In the following section we state the basic setup and definitions. In Section 3.1 we show that for every on-line algorithm, there is a worst case path that yields a cost which is quadratic in l_{opt}. *MRSAM* and *MRBUG* are introduced and their competitiveness is analyzed in Sections 4 and 5 accordingly. It is shown that the length of the path traveled by the robot during execution of *MRSAM* and *MRBUG* is at most quadratic in l_{opt}, implying that up to a constant coefficient *MRSAM* and *MRBUG* have optimal competitiveness. In those sections, *MRSAM* and *MRBUG* are proved to be complete. Simulation results of *MRSAM* and *MRBUG* executions in an office-like environment are described in Section 6. An extension to *MRSAM* algorithm handling multi-target environment is discussed in Section 4.2 along with some practical speedups of the algorithm. Finally, we conclude and discuss additional research directions and future work.

2 Basic setup and definition of competitiveness

Our basic assumptions are as follows. Each mobile robot is a freely moving planar body of size D, where $D > 0$ is a given constant. The mobile robots have any shape contained in a $D \times D$ square. Each robot is equipped with three sensors which are assumed ideal. The first sensor measures the robot's position with respect to a fixed reference frame. The second sensor is an obstacle detection tactile or short range sensor which allows tracing of an obstacle's boundary. The third sensor is a target recognition sensor, note that we assume recognition of the target upon arrival, i.e., when the robot is on top of it or within the detection range. The target detection sensor is also referred to as a covering tool, thus we consider the detection range to contain a square of size D but not to exceed $2D$. The robots use onboard calculation unit, also known as CPU or a micro controller. Communication between the robots is needed at least upon starting and ending of the execution. Thus, the robots must be able to communicate with each other or with a central base station, which serves as a relay. The robots are assumed to have enough memory for the calculations needed and they all move at the same uniform velocity.

Mobile robot motion task's performance can be divided into two parts, physical and computational. Physical performance is generally measured by the time it takes to complete the task, and computational performance is composed of the computational complexity and the memory capacity. The algorithms we develop use memory storage which is at most linear in the size of the environment. This constraint

[2] Implementing the algorithm in a centralized or decentralized manner would affect the average case performance but will not change the upper and lower bounds, see also Section 4.2

may limit the execution in very large and in unbounded environments and should be re-evaluated in future work. We develop algorithms which have polynomial time computational complexity. Since the time to make the next physical step is much greater than the time to calculate it, and according to the previous assumptions regarding the robots' velocities, we use the path length traveled by the robot that reached the target, l, as the main measure of the physical performance. We use the optimal off-line solution, the shortest path, denoted l_{opt}, as the main performance measure to compare with. The following definition generalizes the traditional notion of linear competitiveness to any functional relationship between l and l_{opt}.

Definition 1 (Generalized Competitiveness [10]) An on-line algorithm solving a task P is $f(l_{\text{opt}})$- competitive when l is bounded from above by a scalable function $f(l_{\text{opt}})$ over all instances of P. In particular, $l \leq c_1 l_{\text{opt}} + c_0$ is the traditional linear competitiveness, while $l \leq c_2 l_{\text{opt}}^2 + c_1 l_{\text{opt}} + c_0$ is a quadratic competitiveness, where the c_i's are positive constant coefficients that depend on the robot size D, the number of robots and the geometry of the environment.

We wish to classify the complexity of a task P. The classification is being done by supplying an upper and lower bounds. If these bounds are of the same functional relationship of l_{opt}, this function is considered to be the competitive complexity class of P. The upper bound is associated with a specific algorithm solving the task P in $f(l_{\text{opt}})$-competitiveness. While the lower bound is the lowest upper bound that can be achieved by any on-line algorithm solving P. This notion is made formal in the following definition.

Definition 2 (Competitive Complexity Class [10]) A universal lower bound on the competitiveness of a task P is a lower bound $l \geq g(l_{\text{opt}})$ over all on-line algorithms for P at worst case conditions. If a competitive upper bound $f(l_{\text{opt}})$ and a universal lower bound $g(l_{\text{opt}})$ for P are the same function up to constant coefficients, this function is the competitive complexity class of P.

The competitive complexity class of a task P is thus a pair of lower and upper bounds on the competitive performance of all on-line algorithms for P, such that the two bounds are identical up to constant coefficients. Note that competitive complexity characterizes the task P itself, not any specific algorithm for P. The remainder of the paper characterizes the competitive complexity class of the multi-robot on-line navigation problem.

3 Universal lower bound

3.1 A universal lower bound for an unknown target

In this section we establish a universal lower bound on the competitive complexity for the problem of navigation to a target which is recognized only upon arrival by a group of robots. A difficult environment [10] that serves to establish the lower bound is a disc containing D-width corridors that emanate radially from the start point S (Fig. 1). Initially a small disc centered at S is free of obstacles. At a certain distance

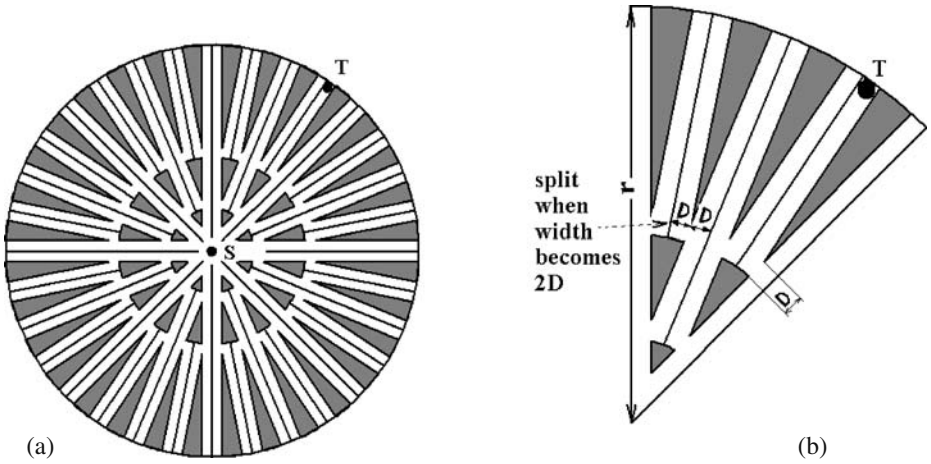

Fig. 1 Gabriely and Rimon [10]. **a** The radial corridors environment. **b** Close up view of the environment

from S eight equally spaced point-size obstacles appear, such that the distance between the obstacles is D (the number eight has no special meaning here). The eight obstacles extend radially as lines and form the boundary of eight passable corridors for the robot. The width of the eight corridors increases as they stretch radially away from S. When the width of a corridor becomes $2D$, the corridor splits into two D-width corridors separated by a cone-shaped obstacle (Fig. 1b). By symmetry all corridors split simultaneously. Hence the cone obstacles that separate corridors just before splitting are truncated at the splitting radius, and become radial lines from this radius onward. Finally, the tip of each cone obstacle is symmetrically "shaved", so that its tip would lie at a distance D away from the truncated cone lying closer to S (Fig. 1b). This shaving allows a D-size robot to enter the two corridors generated by splitting. Note that the shaved off area becomes negligible relative to the cone's total area as the radius of the environment increases. A close inspection of Fig. 1b reveals that the cone obstacles occupy one third of the disc's total area, which is explained in the following section.

In order to show that approximately one third of the environment is occupied by obstacles, we will divide the area into rings, where each ring contains two sub-rings, an inner ring which is free of obstacles and has the width of D, and an outer ring which contains cone obstacles. The cone obstacles in the outer ring have a curve of size $2D - \epsilon$, where ϵ is a small number which appears due to the shaving of the cone tips. As was mentioned before, ϵ becomes negligibly small as the radius of the environment increases. Each outer ring is composed of sections where each section contains one cone obstacle and two D-width corridors of length l (see Fig. 1b). The area of the cone obstacle equals approximately the area of a triangle, $2Dl/2$, and the area of the two corridors equals $2Dl$. It can be seen that the area of the cone is half the area of the free corridors implying that the area of the obstacles approaches one third of the environment as the size of the environment increases. The following lemma is based on the work of [10]. It establishes a quadratic lower bound on the performance of all on line navigation algorithms for a group of robots to a target whose position is recognized only upon arrival.

Lemma 3.1 *Let A be any navigation algorithm for n robots in an unknown planar environment to a target whose position is recognized only upon arrival. Let l be the length of the path generated by A for the robot which found the target, and let l_{opt} be the length of the optimal off-line path. Then l satisfies the quadratic lower bound,*

$$l \geq \frac{4\pi}{3nD}(1-\epsilon)l_{opt}^2$$

where D is the robot size and ϵ is a small positive parameter.

Proof Consider the corridor environment with the target T placed at the end of a distal corridor, at a distance r from S. Since the robots have no knowledge of the environment and has no information where T might lie, they must in worst case inspect every corridor including all distal corridors. (If A is deterministic, we can enforce this worst case scenario by first watching the behavior of A, then placing the target in the last inspected corridor. If A is non-deterministic, we can only guarantee that one outcome of the algorithm would match this worst case scenario.) By construction every distal corridor can be approached from S along a simple radial path. Moreover, the robots must eventually move twice through every corridor of the environment - once in order to inspect a distal corridor and once in order to exit the corridor. An exception to this rule is the last corridor which is inspected once. The total area of the obstacles in the corridor environment is almost one third of the disc area, with the approximation becoming arbitrary close to one third as the disc's radius increases. The total area inspected by the robots is therefore $2\pi r^2/3$. Since all corridors have a width D which is identical to the robots size, the total length of the path traveled by the robots satisfies in worst case the inequality $l_{\text{tot}} \geq 4\pi r^2/3D - r$, where the substraction of r is due to the last corridor which need not be traced backward. In the best case where none of the robots travel any part of the path of the other robots, the total length of the path traveled by one robot satisfies $l \geq 4\pi r^2/3nD - r/n$. Since T is placed at a radial distance r from S we have that $l_{\text{opt}} = r + \epsilon'$, where ϵ' is a small positive parameter which appears due to the transition between successive radial corridors. Substituting for r gives $l \geq (4\pi/3nD)l_{\text{opt}}^2 - l_{\text{opt}}/n - \epsilon'$ .We can write the last inequality as $l \geq l_{\text{opt}}^2(c - (l_{\text{opt}}/n + \epsilon')/l_{\text{opt}}^2)$, where $c = 4\pi/3nD$. Since the quantity $\epsilon = (l_{\text{opt}}/n + \epsilon')/l_{\text{opt}}^2$ can be made arbitrarily small for sufficiently large environments, we obtain the lower bound $l \geq c(1-\epsilon)l_{\text{opt}}^2$. □

3.2 A universal lower bound for a known target

In this section we establish a universal lower bound on the competitive complexity of a group of robots navigating to a known target in an unknown environment. We would like to show that in difficult, close to worst case scenarios, the robots will cover at least a certain area prior to finding the path to the target, and for that purpose we used the environment depicted in Fig. 2. The environment is built from radial corridors which have the width of the robots, D, and one circular corridor containing the target with only one entrance. According to the construction of the environment [11], the area of the obstacles equals one third of the environment's area (Fig. 2b).

The following lemma is based on the work of [10]. It establishes a quadratic lower bound on the performance of all on line navigation algorithms for a group of robots to a known target in an a priori unknown environment.

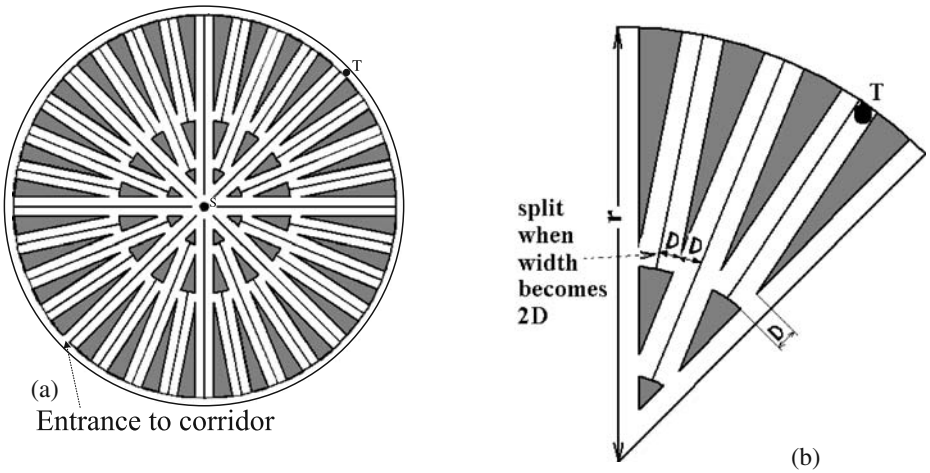

Fig. 2 Gabriely and Rimon [10]. **a** The radial corridors environment. **b** A close-up view of the environment

Lemma 3.2 *Let $\mathcal{AL}$ be any navigation algorithm for $2n$ robots in an unknown planar environment to a target whose position is known. Let l be the length of the path generated by $\mathcal{AL}$ for one of the robots, and let l_{opt} be the length of the optimal off-line path. Then l satisfies the quadratic lower bound,*

$$l \geq \frac{2\pi}{3nD(1+\pi)^2}(1-\epsilon)l_{opt}^2$$

where D is the robot size and ϵ is a small positive number.

Proof Consider the corridor environment with the target $\mathcal{T}$ placed in the outer corridor, at a distance r from $\mathcal{S}$. Since the robots have no knowledge of the environment and has no information where the entrance to the outer corridor might be, they must in worst case inspect every corridor including all distal corridors. If $\mathcal{AL}$ is deterministic, we can enforce this worst case scenario by first watching the behavior of $\mathcal{AL}$, then placing the entrance in the last inspected corridor. If $\mathcal{AL}$ is non-deterministic, we can only guarantee that one outcome of the algorithm would match this worst case scenario. By construction every distal corridor can be approached from $\mathcal{S}$ along a simple radial path. Moreover, the robots must eventually move twice through every corridor of the environment - once in order to inspect a distal corridor and once in order to exit the corridor. An exception to this rule is the last corridor which is considered below. The total area of the obstacles in the corridor environment is almost one third of the disc area, with the approximation becoming arbitrary close to one third as the disc's radius increases. The total area inspected by the robots is therefore $\frac{2\pi r^2}{3}$. Since all corridors have a width D which is identical to the robots size, the total length of the path traveled by the robots satisfies in worst case the inequality $l_{\text{tot}} \geq \frac{4\pi r^2}{3D} - r$,where the substraction of r is due to the last corridor which need not be traced backward. A good algorithm will distribute the

work equally between the robots, in a way that none of the robots will travel any part of the path of the other robots, thus the total length of the path traveled by one robot satisfies $l \geq \frac{2\pi r^2}{3nD} - r$. Since $\mathcal{T}$ is placed in the circular outer corridor, we have that $l_{\text{opt}} \leq (1+\epsilon')(1+\pi)r$, where ϵ' is a small positive number. This is a worst case bound for cases where the target is placed in the circular corridor farthest from the entrance. It follows that $r \geq \frac{l_{\text{opt}}}{(1+\epsilon')(1+\pi)}$. On the other hand, $r \leq l_{\text{opt}}$ in the above mentioned environment. Substituting the last two inequalities into the lower bound on l gives

$$l \geq \frac{2\pi}{3nD}r^2 - r = \frac{2\pi}{3nD(1+\epsilon')^2(1+\pi)^2}l_{\text{opt}}^2 - l_{\text{opt}}.$$

We can write the last inequality as

$$l \geq cl_{\text{opt}}^2\left(\frac{1}{(1+\epsilon')^2} - \frac{1}{cl_{\text{opt}}}\right) = cl_{\text{opt}}^2\left(1-\epsilon'' - \frac{1}{cl_{\text{opt}}}\right),$$

where $c = \frac{2\pi}{3nD(1+\pi)^2}$ and $\frac{1}{(1+\epsilon')^2} = 1-\epsilon''$. Since the quantity $\epsilon = \epsilon'' + \frac{1}{cl_{\text{opt}}}$ contains the quotient D/l_{opt} which can be made arbitrarily small for sufficiently large environments, we obtain the lower bound $l \geq c(1-\epsilon)l_{\text{opt}}^2$. □

4 MRSAM motion algorithm to an unknown target

MRSAM algorithm launches multiple robots from a common starting point S and assigns each robot j to a disc to search for the target T in it, all the discs are concentric and S is their center.

The first robot ($j = 1$) is designated to the initial disc of area S_0, and each of the following robots starts its search in a disc of area larger than the previous disc by a factor of[3] $\alpha > 1$. Namely, the areas of the discs will be S_0,αS_0,$\alpha^2 S_0$,$\alpha^3 S_0 \ldots$. For example, in Fig. 3, robot 1 is initially assigned to search for the target inside a disc of area S_0 and robot 2 is assigned to search inside a disc of area αS_0. After robot 1 completes covering the entire portion of disc 1 which is accessible from S, it starts searching for the target inside disc 3 of area $\alpha^2 S_0$, in this case, robot 1 will find the target while searching in disc 3, before or after robot 2 completed searching in disc 2 and moved on to disc 4 of area $\alpha^3 S_0$. Each robot searches for the target in the accessible portion of the disc allocated to him until the target is detected, or until the entire region accessible from S is explored without finding T.

The search process in each disc is as follows. The robot imposes an on line discretization of the continuous area into a grid of D-size cells [9, 16]. The grid

[3]This is an important property, since the search area must be extended in each step in order to reach the target when the target is positioned outside the first search disc.

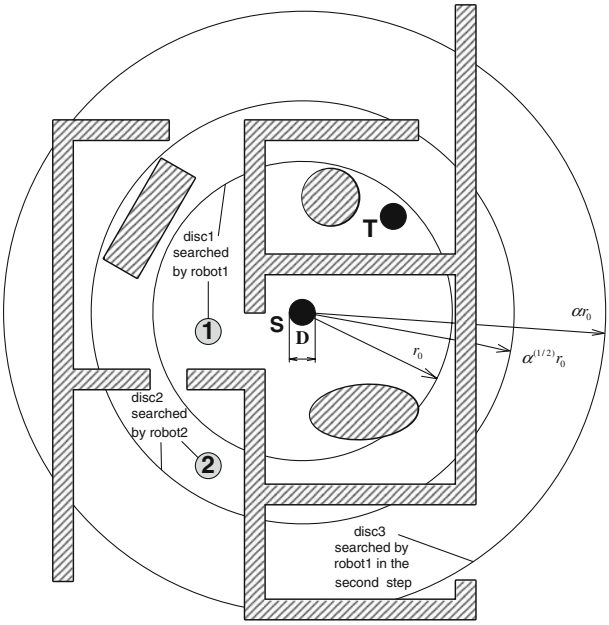

Fig. 3 A group of two robots launched by *MRSAM* searching for the target

consists only of free cells[4] and is surrounded by partially occupied cells. The robot executes a standard area coverage tour on the grid of free cells, while scanning each new cell for the target. Once entering a new cell, the robot additionally scans the neighboring partially occupied cells for T. If the discretization preserves the connectivity of the accessible region (this assumption can be relaxed by a more sophisticated algorithm that monitors local connectivity breakage), then clearly all free and partially occupied cells in the region accessible from S are eventually inspected by the robot [9].

Standard area coverage tour of a grid environment can be implemented in several ways, the most basic one is the DFS (depth first search) algorithm which on-line builds a spanning tree using the grid cells as nodes. The basic idea of DFS is moving in one direction as long as it is possible and when reaching a dead end moving back to a previous node and trying a different direction. A more effective coverage is obtained using the STC (spanning tree covering) algorithm [9]. Here the robot on-line builds a spanning tree for nodes representing $2D$ size cells. While building the tree in a DFS way it circumnavigate it, consequently covering effectively the environment with no repetitions.

The robots cover the area of the discs until they reach the Target. If the target was not detected in the initial disc, the robot is assigned to the next non-occupied disc. A formal description of the algorithm follows.

[4]Partially occupied cells can be treated using Exact STC [8] which ensures a complete coverage by circumnavigating each obstacle encountered during navigation. The Exact STC algorithm generates a path length which is longer than the STC by a constant.

Basic *MRSAM* Algorithm
Sensors: A position sensor.
An obstacle detection sensor.
A target detection sensor.
Input: A start point S.
An initial search radius r_0.
A group of n searching robots.
Initialization:
Set multiplication factor $\alpha = (n+1)^{1/n}$.
For each robot j, $j = 1, \ldots, n$:
Set current search disc $p_j = j$,
Set initial search radius $r_j(p_j) = \alpha^{\frac{p_j - 1}{2}} r_0$.
For each robot j, Repeat:
Execute a *coverage tour* on the grid contained in the disc of radius $r_j(p_j)$ centered at S. Scan each new free cell and its partially occupied neighbor cells for T until one of the following occurs:
(1) The target is reached: STOP.
(2) If no new free cell is encountered during the current coverage tour:
STOP, the target is unreachable.
(3) Else, move to the next unoccupied disc:
Set $p_j = p_j + n$,
Set $r_j(p_j) = \alpha^{\frac{p_j - 1}{2}} r_0$.
End of Repeat loop

Before analyzing the competitiveness of the algorithm, we make the following remarks. First, during the initialization section, after getting the values of n and r_0, each robot can calculate its future search discs and the corresponding radii, which means that the robots does not need to communicate with each other until the rest of the execution, apart from a stopping signal when the target is found. This implies a decentralized approach with no or limited communication. Second, a robot that finished searching a disc will immediately proceed to the next disc assigned to it. The coverage tour in the beginning of each repeat loop can be executed with a trivial DFS algorithm using linear memory [9]. The algorithm's average performance can be improved if each robot covers in each step only those cells which lie in the ring added to the previous disc, this is discussed later in this paper. Fourth, if the target is inaccessible from S, the algorithm would stop only when it has completely covered the connected component of the environment containing S. A detailed example of *MRSAM* execution appears in Section 6.

4.1 Competitive complexity analysis of MRSAM

We now establish an upper bound on the path length of *MRSAM* in terms of l_{opt}. The following proposition establishes a quadratic competitive upper bound on *MRSAM*.

Proposition 4.1 *If the target T is reachable from S,* MRSAM *finds the target using n robots and the path length l_j traveled by the robot which found the target satisfies the quadratic inequality,*

$$l_j < \frac{2\pi\alpha^{n+1}}{D(\alpha^n - 1)} l_{opt}^2 + \frac{2\pi r_j^2(j)}{D}$$

where D is the robot size, $r_j(j) = \alpha^{\frac{j-1}{2}} r_0$ is the first search radius of robot number j, r_0 is the radius of the initial search disc, $\alpha > 1$ is the multiplication factor, and l_{opt} is the length of the optimal off-line path from S to T. Note that the upper bound is scalable, in the sense that both summands have units of length.

Proof First consider the case where the radius of the minimal disc containing l_{opt}, denoted r_{opt}, is greater than the initial radius of the search disc of robot number n, denoted $r_n(n)$. In this case, the initial search disc area is expanded at least once before the target is found. Suppose the search disc is expanded $i-1$ times until the target is found (in disc i). Since the radius of a disc is increased by a factor of $\sqrt{\alpha}$ at each step, its area increases in each step by a factor of α. Let $S_0 = \pi r_0^2$, and let S_j denote the total area of the regions inspected by robot j which found the target after searching in k discs[5] (these areas include free as well as partially occupied cells inspected by the robot.), then S_j is bounded by:

$$\begin{aligned} S_j &\le S(j) + S(j+n) + S(j+2n) + \ldots + S(i) \\ &= \alpha^{j-1} S_0 + \alpha^{j-1+n} S_0 + \alpha^{j-1+2n} S_0 + \ldots + \alpha^{(i-1)} S_0. \end{aligned}$$

Substituting $i = j + n(k-1)$ yields

$$\alpha^{(i-1)} S_0 = \alpha^{j-1+n(k-1)} S_0.$$

And thus we get:

$$\begin{aligned} S_j &\le \alpha^{j-1} S_0 + \alpha^{j-1+n} S_0 + \alpha^{j-1+2n} S_0 + \cdots + \alpha^{j-1+n(k-1)} S_0 \\ &= \alpha^{j-1} S_0 \left[(\alpha^n)^0 + (\alpha^n)^1 + (\alpha^n)^2 + \cdots + (\alpha^n)^{k-1} \right] \\ &= \alpha^{j-1} S_0 \frac{(\alpha^n)^k - 1}{\alpha^n - 1} = S_0 \frac{\alpha^{j-1}(\alpha^n)^k - 1}{\alpha^n - 1} \\ &\le S_0 \frac{\alpha^{j-1}(\alpha^n)^k}{\alpha^n - 1}, \end{aligned}$$

where we used $q = \alpha^{j-1}$, $\lambda = (\alpha^n)$, and $w = k$ in the sum of geometric series formula:

$$q + q\lambda + q\lambda^2 + q\lambda^3 + \ldots + q\lambda^{w-1} = \frac{q(\lambda^w - 1)}{(\lambda - 1)}.$$

[5] k can be considered a global step. In the initial global step the n robots search in the first n consequent discs, in the next global step the robots search in the next n consequent discs and so on.

Since the disc of radius l_{opt} already contains at least one path from S to T, *MRSAM* finds the target in a disc of radius at most $\sqrt{\alpha} l_{\text{opt}}$. It follows that the area of the i^{th} search disc, $\alpha^{i-1} S_0$, satisfies the inequality $\alpha^{i-1} S_0 < \alpha \pi l_{\text{opt}}^2$. Substituting this inequality into the bound on S_j gives:

$$\begin{aligned} S_j &< \frac{\pi \alpha^j (\alpha^n)^k}{(\alpha^n - 1)\alpha^{i-1}} l_{\text{opt}}^2 = \frac{\pi \alpha^j \alpha^{nk}}{(\alpha^n - 1)\alpha^{j+n(k-1)-1}} l_{\text{opt}}^2 \\ &= \frac{\pi \alpha^j \alpha^{nk}}{(\alpha^n - 1)\alpha^j \alpha^{nk} \alpha^{-n} \alpha^{-1}} l_{\text{opt}}^2 = \frac{\pi}{(\alpha^n - 1)\alpha^{-n} \alpha^{-1}} l_{\text{opt}}^2 \\ &= \frac{\pi \alpha}{1 - \alpha^{-n}} l_{\text{opt}}^2 = \frac{\pi \alpha^{n+1}}{\alpha^n - 1} l_{\text{opt}}^2, \\ S_j &< \pi \frac{\alpha^{n+1}}{\alpha^n - 1} l_{\text{opt}}^2. \end{aligned} \tag{1}$$

Now recall that *MRSAM* guides the robot only through free grid cells. The total area of the free grid cells is at most S_j. Hence the total number of grid cells visited by the robot (including repetitive visits), denoted m, is bounded by $m \leq S_j / D^2$. On-line coverage algorithms such as DFS guide the robot along a path whose length in cells is at most twice the total number of grid cells [9]. Since each cell has size D, the total length of the path traveled by the robot is $l_j \leq 2mD$. Thus,

$$l_j \leq 2mD \leq \frac{2}{D} S_j < \frac{2\pi \alpha^{n+1}}{D(\alpha^n - 1)} l_{\text{opt}}^2 = C_n \frac{2\pi}{D} l_{\text{opt}}^2.$$

where we substituted the inequality $S_j < \frac{\pi \alpha^{n+1}}{\alpha^n - 1} l_{\text{opt}}^2$, and $C_n = \frac{\alpha^{n+1}}{\alpha^n - 1}$ is constant per execution since it depends on n alone. Finally, the constant term $\frac{2\pi r_j(j)^2}{D}$ bounds the path length traveled by the robot in the case where $r_{\text{opt}} \leq r_n(n)$. In this case the target is found inside the initial search disc of robot j, whose radius is $r_j(j)$. □

The following lemma, inspired by Baeza-Yates et al. [3], computes the best multiplication factor.

Lemma 4.2 *The Competitive Complexity of* MRSAM *is minimal when the multiplication factor* α *equals* $\alpha = (n+1)^{1/n}$*, where n is the number of the robots in the group.*

Proof Suppose the target was found in the i^{th} disc by robot number j after covering that disc entirely in the worst case scenario. The area of the x^{th} disc is $S(x) = \alpha^{x-1} S_0$ where α is the area multiplying factor and $S(1) = S_0$. The total area S_j covered by that robot as obtained in (4) is:

$$S_j < \pi C_n l_{\text{opt}}^2, \text{ where } C_n = \frac{\alpha^{n+1}}{\alpha^n - 1}.$$

Minimizing S_j for α while taking into account $\alpha > 1$ as explained before, and $n \geq 1$ for realistic execution,

$$\frac{\partial}{\partial \alpha}(C_n) = \frac{\partial}{\partial \alpha}\left(\frac{\alpha^{n+1}}{\alpha^n - 1}\right) = \frac{(n+1)\alpha^n(\alpha^n - 1) - \alpha^{n+1}n\alpha^{n-1}}{(\alpha^n - 1)^2}$$
$$= \frac{\alpha^n(n\alpha^n + \alpha^n - n - 1 - n\alpha^n)}{(\alpha^n - 1)^2} = \frac{\alpha^n(\alpha^n - n - 1)}{(\alpha^n - 1)^2}.$$

Equating with zero for finding extremum point yields,

$$\alpha = (n+1)^{1/n}.$$

This is an extremum value, a second derivative will check the minimality of α,

$$\frac{\partial^2}{\partial \alpha^2}(C_n) = \frac{\partial^2}{\partial \alpha^2}\left(\frac{\alpha^{n+1}}{\alpha^n - 1}\right) = \frac{n\alpha^{n-1}(1 + n + (n-1)\alpha^n)}{(\alpha^n - 1)^3}. \tag{2}$$

Computing the second derivative at $\alpha = (n+1)^{\frac{1}{n}}$ yields,

$$\frac{\partial^2}{\partial \alpha^2}(C_n) = \frac{(n+1)^{2-\frac{1}{n}}}{n} = \frac{(n+1)^2}{n(n+1)^{\frac{1}{n}}} = \frac{(n+1)^2}{n\alpha}.$$

Since $\alpha > 1$, $n \geq 1$ and $\alpha^n - 1 > 0$, All the terms in the righthand side of the equation above are positive and therefore l_j gets minimal values when $\alpha = (n+1)^{1/n}$. □

Corollary 4.3 MRSAM *is complete.*

Proof The first important property established in Proposition 5.4, is that if the target T is reachable, *MRSAM* will find it. The second property is that *MRSAM* will find the target in a finite and limited time and is deduced from the bound on the path length introduced in Proposition 5.4. □

Theorem 1 (Quadratic Competitive Complexity Class) *The online multi-robot navigation problem belongs to the quadratic competitive complexity class.*

Proof A competitive complexity class, as defined in Definition 2, is formed from two bounds, lower and upper bounds on the competitiveness of a task. According to Lemma 3.2 the lower bound of the problem discussed above has a quadratic-competitive complexity and is

$$l \geq \frac{4\pi}{3nD}(1 - \epsilon)l_{\text{opt}}^2 .$$

Since the upper bound of *MRSAM*, as shown in Proposition 5.4, is also quadratic in l_{opt},

$$l_j < \frac{2\pi\alpha^{n+1}}{D(\alpha^n - 1)}l_{\text{opt}}^2 + \frac{2\pi r_0^2}{D} ,$$

this navigation problem belongs to the quadratic competitive complexity class. □

In order to compare the performance of *MRSAM* running more than one robot with the performance of other algorithms running only one robot, we will compare the upper bound on the path length l_j of the robot that found the target for *MRSAM* with multi-robot execution ($n \to \infty$) and for *MRSAM* execution with only one robot ($n = 1$). This is done by calculating C_n for the two cases above. First, for the case where $n \to \infty$, it can be shown that α goes to 1, i.e.,

$$\lim_{n\to\infty} (n+1)^{\frac{1}{n}} = 1 \ ,$$

and thus, C_n approaches 1, as well,

$$\lim_{n\to\infty} C_n = \lim_{n\to\infty} \frac{\alpha^{n+1}}{\alpha^n - 1} \ ,$$

substituting $\alpha = (n+1)^{1/n}$, yields,

$$\lim_{n\to\infty} C_n = \lim_{n\to\infty} \frac{(n+1)^{\frac{n+1}{n}}}{n} = 1 \ .$$

On the other hand, for the second case, where $n = 1$,

$$\alpha = (1+1)^{1/1} = 2,$$

therefore,

$$C_n = \frac{2^{1+1}}{2^1 - 1} = 4,$$

And thus,

$$l < \frac{8\pi}{D} l_{\text{opt}}^2 + \frac{2\pi r_0^2}{D}.$$

The last result coincides with previous results of an optimal algorithm for the same problem with one robot [10]. It can immediately be seen that when $n \to \infty$, *MRSAM* performs 4 times faster than the optimal algorithm which solves the same problem using one robot. It should be noted that for the constraints $\alpha > 1$ and $n \geq 1$ mentioned above, α is a monotonic rising function and thus C_n, is a monotonic rising function, as well. C_n range is $1 < C_n \leq 4$.

Some more values of α and C_n for several cases of n are shown in Table 1. When using one hundred robots, C_n approaches one, and *MRSAM* multiplies the

Table 1 Some α and C_n values corresponding to n entries

No. of robots n	Multiplication factor α	Relative performance C_n	Comparison to single robot $4/C_n$
1	2	4	1
2	1.732	2.598	1.54
4	1.495	1.869	2.14
13	1.225	1.319	3.03
100	1.04	1.058	3.78

performance compared to execution with one robot by a factor of 3.78. Using 4 robots, *MRSAM* doubles the performance and with 13 robots and above it triples the performance compared to one robot execution.

4.2 Extensions and practical speedups of *MRSAM*

Robustness is an important issue in practical situations when completion of the task is a matter of great significance and the units involved tend to fail or malfunction due to hardware or software flaws. *MRSAM* is robust in the sense that $n-1$ out of the n robots can cease to work and yet the target will be found not unboundedly. This robustness is achieved thanks to the fact that each consequent disc is larger and contains the previous disc, such that if the setup includes communication between the robots, *MRSAM* can adjust itself each time a robot fails by recalculating the new multiplication factor α according to the new number of functioning robots, n, and if there is no communication between the robots, each robot will continue its original search plan and eventually find the target, producing a longer path length.

***MRSAM* extended to deal with multiple targets** The basic *MRSAM* Algorithm is designed to find a single target whose position is unknown in an unknown environment. The algorithm can easily be extended to solve a similar problem with more than one target for the following cases: When dealing with a finite number of targets, changing the algorithm is done by removing the direction to the robots to STOP after reaching the first target (step (1)), and adding a target counter and a direction to STOP after that counter has reached a specific value. When the number of targets is unlimited, some other restriction must be made in order to restrain the range of search. That restriction is done by limiting i, the number of steps the algorithm should perform before terminating.

Proposition 4.4 *Let k targets T_i, $i = 1, 2, \ldots, k$ be reachable from S, then* MRSAM *finds all targets using n robots and the path length l_j traveled by the robot which found the last target satisfies the quadratic inequality,*

$$l_j < \frac{2\pi\alpha^{n+1}}{D(\alpha^n - 1)} l_{opt}^2 + \frac{2\pi r_j^2(j)}{D} \tag{3}$$

where D is the robot size, $r_j(j) = \alpha^{\frac{j-1}{2}} r_0$ is the first search radius of robot number j, r_0 is the radius of the initial search disc, $\alpha > 1$ is the multiplication factor, and l_{opt} is the length of the optimal off-line path from S to the last target, T_k. Note that the upper bound is scalable, in the sense that both summands have units of length.

Proof The farthest target T_k in the sense of a path length from S should be reached prior to termination. According to Proposition 5.4, the path length l_j traveled by the robot which found the last target satisfies the quadratic inequality (3). Note that all other targets are closer to start, S, the robots cover the entire discs area, and

the targets counter will prevent stopping before reaching the last target. Therefore, targets $T_1, ..., T_{k-1}$ will be found prior to finding target T_k. The last target T_k in the multi target execution corresponds to the target T in the basic *MRSAM* algorithm, thus the upper bound holds. □

Practical speedup The individual robot in the basic *MRSAM* algorithm is assigned to a disc in which it performs a *coverage tour* searching for the target. The average performance of the algorithm can be improved by directing each robot to search only in the ring added to the last disc that has already been searched by itself or by one of the other robots, and that will be done only if the previous disc connected area was totaly covered, otherwise, the robot will perform a full search on the whole disc.

The case where a robot needs to "return" to a partially uncovered previous disc, can be improved by means of a common environment map which is updated and shared by all the robots. That way, the robot that needs to make a full disc cover will be able to calculate the shortest path to return to the area it did not cover. Creating and maintaining a common geometric map of the environment that includes information about the area that was already covered can save excess searching for the robots and thus speedup the average performance of *MRSAM*. Such extension requires communication capabilities.

Communication usually implies a centralized solution, but can also be implemented by a decentralized solution if the communication equipment allows it, e.g., radio transmitting power, line-of-sight communication, etc. Centralized systems rely on a single controlling unit, which computes the next steps for all the robots, allowing them to have weaker and cheaper processors and memory units. The practical speedups dealt in this section will enhance the performance in the average case, however, worst case scenarios will imply upper bound executions.

5 MRBUG motion algorithm to a known target

MRBUG, a new algorithm solves the problem of finding a path to a known target using a group of robots. *MRBUG* uses another new algorithm, *PBUG1* for each pair of robots it deploys as a sub procedure in the main algorithm. *PBUG1* algorithm is described in the following section.

5.1 *PBUG1* motion algorithm for a pair of robots

We now introduce *PBUG1*, a version of *BUG1* [22] for a pair of robots which uses the same problem's definitions as *BUG1*. In *PBUG1* a pair of robots that start from a common start point $\mathcal{S}$ needs to find a path to a target $\mathcal{T}$ whose position is known in an unknown planar environment. The pair of robots will move together toward the target in a straight line until they hit an i^{th}obstacle at a point marked as *Hit point* H^i, $i = 1, 2, ...$. At that point they split, robot R_{L} turns left and robot R_{R} turns right, and they circumnavigate the obstacle from different directions (Fig. 4). On that account, each robot encircle half of the obstacle's perimeter. While moving, each robot calculates and remembers the closest point on the obstacle's boundary to the

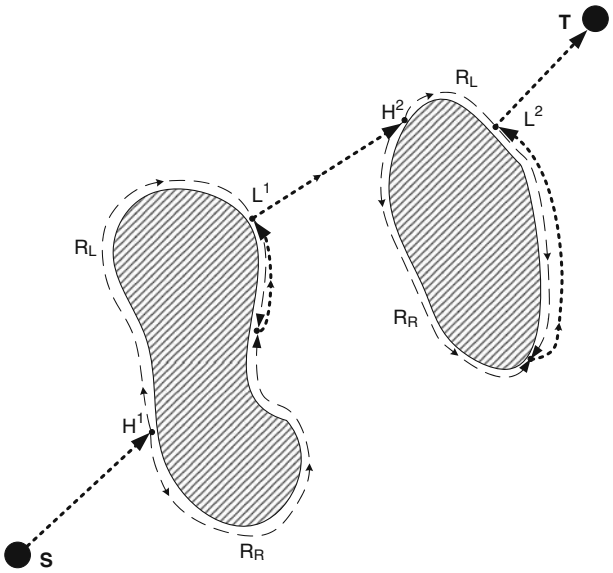

Fig. 4 A pair of two robots, R_L and R_R executing *PBUG1*. The *dense dashed lines* mark a mutual path of the two robots

target. Upon meeting, the robots compare the recorded information, decide which point is the closest to the target, join and again move together to that closest point which they mark as *Leave point* L^i, $i = 1, 2, \ldots$. Finally, the robots continue to move together toward the target.

5.1.1 Setup and definitions of PBUG1

The basic setup and definitions of *BUG1* in section 2 of [22] for one robot are applicable in *PBUG1* for two robots along with a few modifications and adjustments as follows. The Automaton of [22] is extended in *PBUG1* into two mobile robots, R_L and R_R, each has a different pre defined local direction for moving around an obstacle, left and right accordingly. The hit and leave points are common for both of the robots. The procedure *PBUG1* needs only one register for each robot, *Reg*1, which is used to store the coordinates of the current point, Q_m, of the minimum distance between the obstacle's boundary and the target. The original *Reg*2 and *Reg*3 registers from *BUG1* are not needed here, since the robots do not need to calculate the direction of movement to the next leave point, they just compare their Q_m points and go together to the one with the shorter value.

5.2 Target reachability test

PBUG1 determines that the target is unreachable and trapped inside an obstacle using *BUG1* method [22], which checks the direction to the target after circumnavigating an obstacle. If this direction points into the last obstacle, the target is surrounded by that obstacle, since the leaving point is the closest point to the target on the obstacle's boundary. A formal description follows.

5.2.1 A formal description of PBUG1 algorithm

***PBUG1* Algorithm**
Sensors: A position and orientation sensors.
An obstacle detection sensor.
Input: Position of a start $\mathcal{S}$ and a target $\mathcal{T}$.
A pair of robots: R_{L}, R_{R}.
Initialization: For each of the robots in the pair R_{L}, R_{R}:
Define local direction: *Left* for R_{L}, *Right* for R_{R}.
Set i=1.
Set initial leave point $L^0 = \mathcal{S}$.
For each of the two robots R_L, R_R, Repeat:
From the point L^{i-1}, move toward the target along a straight line
until one of the following occurs:
(1) The target is reached: STOP.
(2) An obstacle is encountered: Define a hit point H^i.

Turn in the direction of the predefined local direction and follow the obstacle's boundary according to that direction. While circumnavigating the obstacle, calculate and record the coordinates of the closest point to the target, $Q_{\mathrm{m_L}}$ and $Q_{\mathrm{m_R}}$ for R_{L} and R_{R} respectively, until one of the following occurs:

(a) The target is reached: STOP.
(b) Upon meeting each other, exchange

information and calculate the closest point to $\mathcal{T}$: $Q_{\mathrm{m}} = \min(Q_{\mathrm{m_L}}, Q_{\mathrm{m_R}})$.
Define a new leave point $L^i = Q_{\mathrm{m}}$.
Apply the test for target reachability:
(i) If the target is not reachable: STOP.
(ii) Else, move to L^i: If $Q_{\mathrm{m}} = Q_{\mathrm{m_L}}$, trace back
R_{L} path. Otherwise, trace back R_{R} path.
Set $i = i + 1$.

End of Repeat loop

5.3 *MRBUG* algorithm for a group of robots

MRBUG algorithm launches n pairs of robots from a common starting point $\mathcal{S}$ and assigns each pair j' to a different ellipse to search for a path to the target $\mathcal{T}$ in it, each ellipse's focal points are $\mathcal{S}$ and $\mathcal{T}$.

The first pair of robots ($j' = 1$) is designated to the initial ellipse of area A_0, and each of the following robots starts its search in an ellipse of area larger than the previous ellipse's area by a factor of[6] $\alpha > 1$, namely, the areas of the ellipses will be

[6]This is an important property, since the search area must be extended in each step in order to reach the target when the path to the target is positioned outside the first search ellipse.

A_0,αA_0,$\alpha^2 A_0$,$\alpha^3 A_0 \ldots$. For example, in Fig. 5, robots 1_{L} and 1_{R} are initially assigned to search for a path to the target inside an ellipse of area A_0 and robots 2_{L} and $_{\mathrm{R}}$ are assigned to search inside an ellipse of area αA_0. The search for the path inside an ellipse is done by the pair of robots assigned to that ellipse using *PBUG1*.

In *MRBUG*, the execution of *PBUG1* regards the ellipse as a virtual obstacle's boundary. If the target is detected, the algorithm terminates, otherwise, the pair of robots repeats the process on the next unassigned ellipse in the series. A formal description of the basic algorithm follows.

5.3.1 A formal description of MRBUG algorithm

***MRBUG* Algorithm**
Sensors: A position and orientation sensors.
An obstacle detection sensor.
Input: Position of a start $\mathcal{S}$ and a target $\mathcal{T}$ points.
An initial ellipse with focal points $\mathcal{S}$ and $\mathcal{T}$ and area A_0.
n pairs of robots $\{1_L, 1_{\mathrm{R}}, 2_{\mathrm{L}}, 2_{\mathrm{R}}, \ldots, n_L, n_R\}$.
Initialization: For each robot pair j', $j' = 1, \ldots, n$:
Set current search ellipse $e_{j'} = j'$,
Set initial leave point $L^0_{e_{j'}} = \mathcal{S}$,
Set multiplication factor $\alpha = (n+1)^{1/n}$,
Set initial search area $A_{j'}(e_{j'}) = \alpha^{e_{j'}-1} A_0$.
For each robot pair j', Repeat:
Initialize *PBUG1* with the following parameters:
Create an outer virtual obstacle's boundary with an ellipse of area $A_{j'}(e_{j'})$ having focal points $\mathcal{S}$ and $\mathcal{T}$.
Start point is $\mathcal{S}$, target is $\mathcal{T}$.
Set $i = 1$.
Leave point is $L^0_{e_{j'}}$.
Execute *PBUG1* until one of the following occurs:
(1) *PBUG1* terminates at $\mathcal{T}$: STOP, target is found.
(2) $\mathcal{T}$ is trapped inside an obstacle:
(a) If the obstacle does not intersect the $e_{j'}$ ellipse: STOP, the target is unreachable.
(b) Else, move to the next unoccupied ellipse:
Set $e_{j'} = e_{j'} + n$.
Set $L^0_{e_{j'}}$ at *PBUG1* termination point.
Set $A_{j'}(e_{j'}) = \alpha^{e_{j'}-1} A_0$.
End of Repeat loop

Before analyzing the competitiveness of the algorithm, we make the following remarks. First, during initialization, after getting the values of n and A_0, each robot is assigned to a number j' and to a local direction, *Left* or *Right* and thus can calculate its future search ellipses and corresponding areas, which means that after a pair of

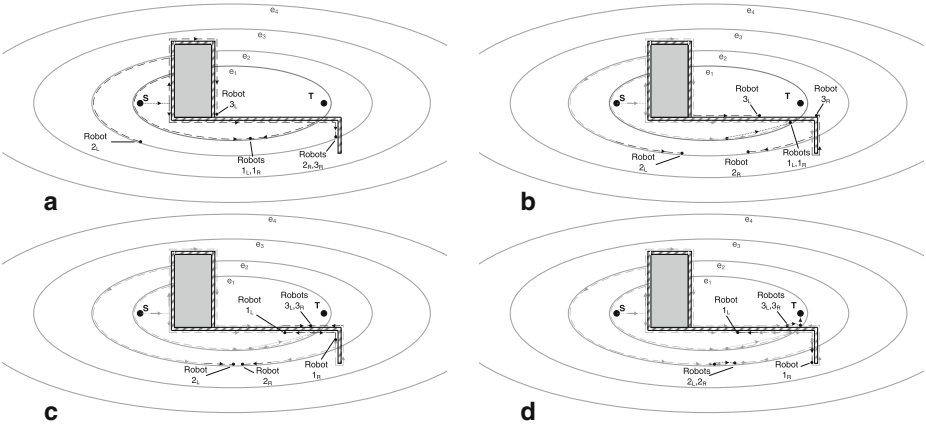

Fig. 5 a–d Execution example of *MRBUG* with three pairs of robots. The *dense dashed lines* indicates a mutual path of a pair. The path traversed in the previous step is colored in *grey*

robots has finished searching for a path in an ellipse, it can immediately continue to search in the next ellipse regardless of the state of the other robots. Communication and decentralization are treated later in Section 5.3.5. Second, the method *PBUG1* uses to determine that the target is unreachable and trapped inside an obstacle in step 2 is discussed in Section 5.2. *MRBUG* assures in step 2a that the robots were not bounded by the ellipse and thus guarantee that the target is unreachable. Third, regarding the memory requirements, in *MRBUG*, each robot executing *PBUG1* uses the same amount of memory as in *BUG1* with a little modification, plus a constant amount of memory. Namely, the target position $\mathcal{T}$, the current obstacle's hit point, the distance of the closest point on the current obstacle's boundary to the target and the two distances to that point along the obstacles' boundaries from its current position are *BUG1* necessities. *PBUG1* memory modification is in the last requirement, where these two distances to the leave point are not necessary and should be recorded by each robot, as was discussed in Section 5.1.1. *MRBUG* additional memory requirements are the start position $\mathcal{S}$ and the current ellipse's area $A_{\hat{f}}(e_{\hat{f}})$.

5.3.2 Execution example of MRBUG

In the following example depicted in Fig. 5, *MRBUG* launches three pairs of robots, $1, 2, 3$ to search for a path to the target in a very simple environment. Each robot pair is initially assigned to a bounding ellipse, e_1, e_2, e_3 to execute *PBUG1* in it, and each robot in a pair is assigned to a different local direction, $Left$, $Right$: $1_L, 1_R, 2_L, 2_R, 3_L, 3_R$. At first, the robot pairs are moving directly toward the target, and as they encounter an obstacle they split, and each robot moves in its local direction. It can be observed that a part of the path is traversed by all the robots together at the same time, and the robots will move together as long as they are within the boundary of the first ellipse. As depicted in Fig. 5a, while robot pair no. 2 is traversing the second ellipse, robot pair no.1 finishes traversing the obstacle's boundary which lies inside the first bounding ellipse and the ellipse itself. It can be seen that robot pair no. 3 does not have to traverse ellipse no. 3 in this example,

since it does not intersects the obstacle. After meeting each other, the robots of pair no. 1 move toward their first leave point which is the closest point to the target they encountered in their traverse, there they conclude that they cannot reach the target from ellipse no. 1 (Fig. 5b). Thus, the robots of pair no. 1 start executing *PBUG1* in ellipse no. 4. Meanwhile, robot pair no. 3 meets on the obstacle's boundary as can be seen in Fig. 5c. While robot pair no. 1 is busy with its search in ellipse no. 4, robot pair no. 2 meet on ellipse no. 2, and afterward move together toward the closest point to the target .At that time interval robot pair no. 3 move together to its leave point on the obstacle's boundary and from there it continues without any more obstacles in its way and reach the target (Fig. 5d).

5.3.3 Competitive complexity analysis of MRBUG

We now establish an upper bound on the path length of the robot that reached the target while executing *MRBUG* in terms of the optimal off-line solution, l_{opt}. In order to calculate the length of the path generated by a robot of size D tracing a $\mathcal{B}_i$ obstacle's boundary, we first substitute the obstacle into an object $\boldsymbol{\mathcal{B}_i}$ in a way that changes its area and perimeter, but preserves the path length property mentioned above. In the following two lemmas we use terms related to *Configuration Space*. The configuration space (or $\mathcal{C}$-space) of a disc shaped robot is $\mathbb{R}^2$, and the $\mathcal{C}$-space obstacle $\mathcal{CB}_i$ consist of all robot configurations where it intersects the obstacle $\mathcal{B}_i$.

Definition 3 (Gabriely and Rimon [11]) Let $\mathcal{CB}_i$ be the $\mathcal{C}$-space obstacle induced by an obstacle $\mathcal{B}_i$ for a disc robot of size D. The *traceable obstacle* induced by $\mathcal{B}_i$, denoted $\boldsymbol{\mathcal{B}_i}$, is obtained by filling any internal holes in $\mathcal{CB}_i$ and then shrinking $\mathcal{CB}_i$ inward by a distance of $D/2$.

Lemma 5.1 (Gabriely and Rimon [11]) *Let a planar environment contain z disjoint traceable obstacles $\boldsymbol{\mathcal{B}_i}$, $i = 1, \ldots, z$. Let a disc robot of size D trace the i^{th} obstacle's boundary, and let q_i be the total area swept by the robot during tracing of the i^{th} boundary. let $\mathcal{C}$ be any simple closed curve which surrounds the z regions swept by the robot. Then $\sum_{i=1}^{z} q_i \leq 4\mathcal{A}(\mathcal{C})$, where $\mathcal{A}(\mathcal{C})$ is the area of the traceable obstacle-free points enclosed by $\mathcal{C}$.*

Note that the regions swept during tracing of the individual boundaries may overlap, so that in general the sum $\sum_{i=1}^{z} q_i$ may be larger than $\mathcal{A}(\mathcal{C})$.

Lemma 5.2 *The length l_i of the path traveled by each robot of the pair traversing the i^{th} ellipse is bounded by*

$$l_i \leq 4\frac{A(i)}{D} + \left(||L_i^0 - \mathcal{T}|| - ||L_{i+n}^0 - \mathcal{T}||\right),$$

where $A(i)$ is the area of the i^{th} ellipse, D is the size of each robot, L_i^0 is the start point at the i^{th} ellipse, n is the number of robots, L_{i+n}^0 is the start point at the next ellipse of the pair of robots, and $||\beta - \gamma||$ denotes the Euclidean distance between β and γ.

Proof When moving toward the target in the e_j ellipse, each robot of the pair of robots assigned by *MRBUG* to that ellipse is executing *PBUG1*. The regions swept by the robots during circumnavigation of obstacles in this ellipse (including the ellipse

itself) are surrounded by the ellipse's boundary. Identifying the latter boundary with the curve $\mathcal{A}(\mathcal{C})$ from Lemma 5.1, the total path length of the two robots during circumnavigation of the obstacles is at most $4A(i)/D$, where $A(i)$ is the i^{th}ellipse's area. Since each robot travel exactly half of the way, the path length of one robot is not more than $2A(i)/D$. Recall now that under *BUG1* the robot circumnavigates the boundary of each obstacle at most 1.5 times, here each of the two robots will circumnavigates the boundary of each obstacle only one time at most. Hence, the total length of each robot's path during boundary following is at most $4A(i)/D$. Recall, too, that under *BUG1* motion between obstacles is always directly to the target. The total length of these motion segments equals to the net decrease of the distance of the robot from $\mathcal{T}$, which is $||L_i^0 - \mathcal{T}|| - ||L_{i+n}^0 - \mathcal{T}||$. Adding the two terms gives the path length bound. □

The following lemma states that the last ellipse area is bounded from above. This lemma and its proof is inspired by [11].

Lemma 5.3 *Let $\mathcal{T}$ be reachable from $\mathcal{S}$. If the initial ellipse contains no path from $\mathcal{S}$ to $\mathcal{T}$,* MRBUG *reaches the target in an ellipse whose area $A(i)$ is bounded by*

$$A(i) < \frac{\pi\alpha}{4} l_{opt} \sqrt{l_{opt}^2 - ||\mathcal{S} - \mathcal{T}||^2},$$

where l_{opt} is the length of the optimal off-line path from $\mathcal{S}$ to $\mathcal{T}$, and α is the multiplication factor.

Proof An ellipse with focal points $\mathcal{S}$ and $\mathcal{T}$ satisfies the inequality

$$\varepsilon = x : ||x - \mathcal{S}|| + ||x - \mathcal{T}|| \leq 2a$$

where $2a$ is the length of the ellipse's major axis. Consider now the optimal off-line path from $\mathcal{S}$ to $\mathcal{T}$ of length l_{opt}. Every point x along this path satisfies the inequality $||x - \mathcal{S}|| + ||x - \mathcal{T}|| \leq l_{\text{opt}}$. It follows that the entire optimal off-line path lies in an ellipse with focal points $\mathcal{S}$ and $\mathcal{T}$ and major axis $2a \leq l_{\text{opt}}$. Next recall that the area of an ellipse is given by πab, where $2b$ is the length of the ellipse's minor axis. In our case $||\mathcal{S} - \mathcal{T}||^2/4 + b^2 = a^2$. Hence, $b \leq \frac{1}{2}\sqrt{l_{\text{opt}}^2 - ||\mathcal{S} - \mathcal{T}||^2}$, where we used the inequality $a \leq l_{\text{opt}}/2$. Let $A_{\min}$ denote the area of the smallest ellipse with focal points $\mathcal{S}$ and $\mathcal{T}$ which contain the optimal off-line path. Substituting the expressions for a and b in πab gives the upper bound: $A_{\min} = \pi ab \leq \frac{\pi}{4} l_{\text{opt}} \sqrt{l_{\text{opt}}^2 - ||\mathcal{S} - \mathcal{T}||^2}$. By assumption the initial ellipse contain no path from $\mathcal{S}$ to $\mathcal{T}$. Hence, *MRBUG* multiplies the area of the search ellipse by a factor of αat least once. The area $A(i)$ of the last ellipse searched by *MRBUG* satisfies the inequality $A(i) < \alpha A_{\min}$. Otherwise the previous ellipse has an area $A(i-1) \geq A_{\min}$, which implies that *MRBUG* terminated while inspecting the previous ellipse. Substituting for $A_{\min}$ in the inequality $A(i) < \alpha A_{\min}$ gives the result.

The following proposition establishes a quadratic competitive upper bound on *MRBUG*.

Proposition 5.4 *If the target $\mathcal{T}$ is reachable from $\mathcal{S}$,* MRBUG *finds the target using n robots and the path length l traveled by the robot which reached the target satisfies the quadratic inequality,*

$$l \leq \frac{\pi}{D}\frac{\alpha^{n+1}}{\alpha^n - 1} l_{opt}^2 + ||\mathcal{S} - \mathcal{T}|| + 4\frac{A_0}{D}$$

where D is the robot size, and l_{opt} is the length of the optimal off-line path from $\mathcal{S}$ to $\mathcal{T}$. Note that the upper bound is scalable, in the sense that both summands have units of length.

Proof First consider the case where the initial search ellipse area is expanded at least once before the target is found. Suppose the search ellipse is expanded $i-1$ times until the target is reached (in ellipse i). The ellipses area increases in each step by a factor of α. Let $A_{j'}$ denote the total area of the regions inspected by a robot from pair j' which reached the target after searching in k ellipses[7] (these areas include free as well as partially occupied cells inspected by the robot.), then $A_{j'}$ is bounded by:

$$\begin{aligned} A_{j'} &\leq A\left(j'\right) + A\left(j'+n\right) + A\left(j'+2n\right) + \ldots + A(i) \\ &= \alpha^{j'-1}A_0 + \alpha^{j'-1+n}A_0 + \alpha^{j'-1+2n}A_0 + \ldots + \alpha^{(i-1)}A_0. \end{aligned}$$

Substituting $i = j' + n(k-1)$ yields

$$\alpha^{(i-1)}A_0 = \alpha^{j'-1+n(k-1)}A_0.$$

And thus we get:

$$\begin{aligned} A_{j'} &\leq \alpha^{j'-1}A_0 + \alpha^{j'-1+n}A_0 + \alpha^{j'-1+2n}A_0 + \cdots + \alpha^{j'-1+n(k-1)}A_0 \\ &= \alpha^{j'-1}A_0\left[(\alpha^n)^0 + (\alpha^n)^1 + (\alpha^n)^2 + \cdots + (\alpha^n)^{k-1}\right] \\ &= \alpha^{j'-1}A_0\frac{(\alpha^n)^k - 1}{\alpha^n - 1} = A_0\frac{\alpha^{j'-1}(\alpha^n)^k - 1}{\alpha^n - 1} \\ &\leq A_0\frac{\alpha^{j'-1}(\alpha^n)^k}{\alpha^n - 1}, \end{aligned}$$

where we used $y = \alpha^{j'-1}$, $\lambda = (\alpha^n)$, and $w = k$ in the formula:

$$y + y\lambda + y\lambda^2 + y\lambda^3 + \ldots + y\lambda^{w-1} = \frac{y(\lambda^w - 1)}{(\lambda - 1)}.$$

[7] k can be considered a global step. In the initial global step the n robots search in the first n consequent ellipses, in the next global step the robots search in the next n consequent ellipses and so on.

According to Lemma 5.3, the area of the i^{th} ellipse, $\alpha^{(i-1)}A_0$, satisfies the inequality $\alpha^{(i-1)}A_0 < \frac{\pi\alpha}{4} l_{\text{opt}}\sqrt{l_{\text{opt}}^2 - ||\mathcal{S}-\mathcal{T}||^2} \leq \frac{\pi\alpha}{4} l_{\text{opt}}^2$. Substituting this into the bound on $A_{j'}$ gives

$$\begin{aligned} A_{j'} &< \frac{\pi\alpha^{j'}(\alpha^n)^k}{4(\alpha^n-1)\alpha^{i-1}} l_{\text{opt}}^2 = \frac{\pi\alpha^{j'}\alpha^{nk}}{4(\alpha^n-1)\alpha^{j'+n(k-1)-1}} l_{\text{opt}}^2 \\ &= \frac{\pi\alpha^{j'}\alpha^{nk}}{4(\alpha^n-1)\alpha^{j'}\alpha^{nk}\alpha^{-n}\alpha^{-1}} l_{\text{opt}}^2 = \frac{\pi}{4(\alpha^n-1)\alpha^{-n}\alpha^{-1}} l_{\text{opt}}^2 \\ &= \frac{\pi\alpha/4}{1-\alpha^{-n}} l_{\text{opt}}^2 = \frac{\pi\alpha^{n+1}/4}{\alpha^n-1} l_{\text{opt}}^2 \\ A_{j'} &< \frac{\pi}{4}\frac{\alpha^{n+1}}{\alpha^n-1} l_{\text{opt}}^2 \end{aligned} \tag{4}$$

Hence, the total length of the path traveled by the robot that reached the target is bounded by

$$\begin{aligned} l &= l_{j'} + l_{j'+n} + l_{j'+2n} + \ldots\ldots + l_{j'+(k-1)n} = \sum_{i=1}^{k} l_{j'+(i-1)n} \\ &\leq \frac{4}{D} A_{j'} + \sum_{i=1}^{k} \left(\left|\left| L^0_{e_{j'+(i-1)n}} - \mathcal{T} \right|\right| - \left|\left| L^0_{e_{j'+in}} - \mathcal{T} \right|\right| \right) \end{aligned}$$

Substituting $L^0_{e_{j'}} = \mathcal{S}$ and $L^0_{e_{j'+kn}} = \mathcal{T}$ in the last results we get:

$$l < \frac{\pi}{D}\frac{\alpha^{n+1}}{\alpha^n-1} l_{\text{opt}}^2 + ||\mathcal{S}-\mathcal{T}||$$

Finally, the constant additive term $2A_0/D$ bounds the path length traveled by the robot in case the target is reached from within the first search ellipse. □

The following lemma, inspired by [3], asserts that search area multiplying is indeed an optimal strategy.

Lemma 5.5 *The Competitive Complexity of* MRBUG *is minimal when the multiplication factor* α *equals* $\alpha = (n+1)^{1/n}$.

Proof Let n be the number of robot pairs searching for a path to the target, and suppose the path to the target was found in the i^{th} ellipse by robot pair number j'. The area of the i^{th} ellipse is $A(i) = \alpha^{i-1}A_0$ where α is the area multiplying factor and $A(1) = A_0$. The total area $A_{j'}$ covered by that robot as obtained in (4) is:

$$A_{j'} < \frac{\pi}{4} B_n l_{\text{opt}}^2, \text{ where } B_n = \frac{\alpha^{n+1}}{\alpha^n-1}$$

Minimizing for α while taking into account $\alpha > 1$ as explained before, and $n \geq 1$ for realistic execution,

$$\frac{\partial}{\partial\alpha}(B_n) = \frac{\partial}{\partial\alpha}\left(\frac{\alpha^{n+1}}{\alpha^n-1}\right) = \frac{\alpha^n(\alpha^n-n-1)}{(\alpha^n-1)^2}$$

Equating with zero yields

$$\alpha = (n+1)^{1/n}.$$

This is an extremum value, a second derivative will check the minimality of α,

$$\begin{aligned}\frac{\partial^2}{\partial\alpha^2}(B_n) &= \frac{\partial^2}{\partial\alpha^2}\left(\frac{\alpha^{n+1}}{\alpha^n-1}\right)\\ &= \frac{(\alpha^{2n}-\alpha^n-n\alpha^n)(2n\alpha^{2n-1}-2n\alpha^{n-1})}{(\alpha^n-1)^4}\end{aligned} \tag{5}$$

Since $\alpha > 1, n \geq 1$, which inplies $\alpha^n - 1 > 0$, the denominator of (5) is always positive, thus the numerator determines the sign. Let E denote the numerator. Simplification of E yields, $E = 2n\alpha^{2n-1} + \left(n^2+n\right)\alpha^{n-1}\left(\alpha^{2n}-1\right)$ Since $\alpha > 1, n \geq 1$ and $\alpha^n - 1 > 0$, All the terms in the righthand side of the equation above are positive and therefore $E > 0$, which implies that l_j' gets minimal values when $\alpha = (n+1)^{1/n}$. □

Corollary 5.6 MRBUG *is complete.*

Proof The first important property established in Proposition 5.4, is that if the target $\mathcal{T}$ is reachable, *MRBUG* will find a path to it. The second property is that *MRBUG* will find that path in a finite and limited time and is deduced from the bound on the path length introduced in Proposition 5.4.

Next we wish to show how much a group of robots will improve the algorithm overall path length. In order to compare the performance of *MRBUG* with the performance of algorithms running only one robot, we will compare the upper bound on the path length l_j' of the robot that found the target for *MRBUG* with multi-robot execution ($n \to \infty$) with an execution of an optimal algorithm using one robot (*CBUG*). This is done by calculating B_n. First, for the case where $n \to \infty$, it can easily be shown that α goes to 1,

$$\lim_{n\to\infty}(n+1)^{\frac{1}{n}} = 1$$

and thus, B_n approaches 1, as well.

$$\lim_{n\to\infty} B_n = \lim_{n\to\infty}\frac{\alpha^{n+1}}{\alpha^n-1} = \lim_{n\to\infty}\frac{n+1}{n} = 1$$

and thus

$$l \leq \frac{\pi}{D}l_{\text{opt}}^2 + ||\mathcal{S}-\mathcal{T}|| + 4\frac{A_0}{D}$$

On the other hand, for the second case,

$$l \leq \frac{6\pi}{D}l_{\text{opt}}^2 + ||\mathcal{S}-\mathcal{T}|| + \frac{6A_0}{D}$$

It can immediately be seen that when $n \to \infty$, *MRBUG* performs 6 times faster than the optimal algorithm which solves the same problem using one robot. It should be

noted that for the constraints $\alpha > 1$ and $n \geq 1$ mentioned above, α is a monotonic rising function and thus B_n is a monotonic rising function, as well.

Some more values of α and B_n for several cases of n are shown in Table 2. When using one hundred robot pairs, B_n approaches one, and *MRBUG* multiplies the performance compared to execution with one robot by a factor of 3.78. Using 4 robot pairs, *MRBUG* doubles the performance and with 13 robot pairs and above it triples the performance compared to one robot execution.

5.3.4 Optimality of MRBUG

Using the definitions of competitive complexity, the universal lower bound and the upper bound found in the previous sections, we can now establish a competitive complexity class for the problem solved by *MRBUG* and thus obtain the optimality of *MRBUG*.

Theorem 2 *Quadratic competitive complexity class.*
The on-line multi-robot navigation problem belongs to the quadratic competitive complexity class.

Proof A competitive complexity class, as defined in Definition 2, is formed from two bounds, lower and upper bounds on the competitiveness of a task. According to Lemma 3.2 the lower bound of the problem discussed above has a quadratic-competitive complexity and is

$$l \geq \frac{2\pi}{3nD(1+\pi)^2}(1-\epsilon)l_{\text{opt}}^2$$

Since the upper bound of *MRBUG*, as demonstrated in Proposition 5.4, is also quadratic in l_{opt},

$$l \leq \frac{\pi}{D}\frac{\alpha^{n+1}}{\alpha^n - 1}l_{\text{opt}}^2 + ||\mathcal{S} - \mathcal{T}|| + 4\frac{A_0}{D}$$

this navigation problem belongs to the quadratic competitive complexity class. □

The last theorem exhibits the quadratic competitiveness class of the problem solved by *MRBUG*. Since *MRBUG* has a quadratic competitiveness, *MRBUG* is optimal.

Table 2 Some α and B_n values corresponding to n entries

No. of robots $2n$	Multiplication factor α	Relative performance B_n
2	2	4
4	1.732	2.598
8	1.495	1.869
26	1.225	1.319
200	1.04	1.058

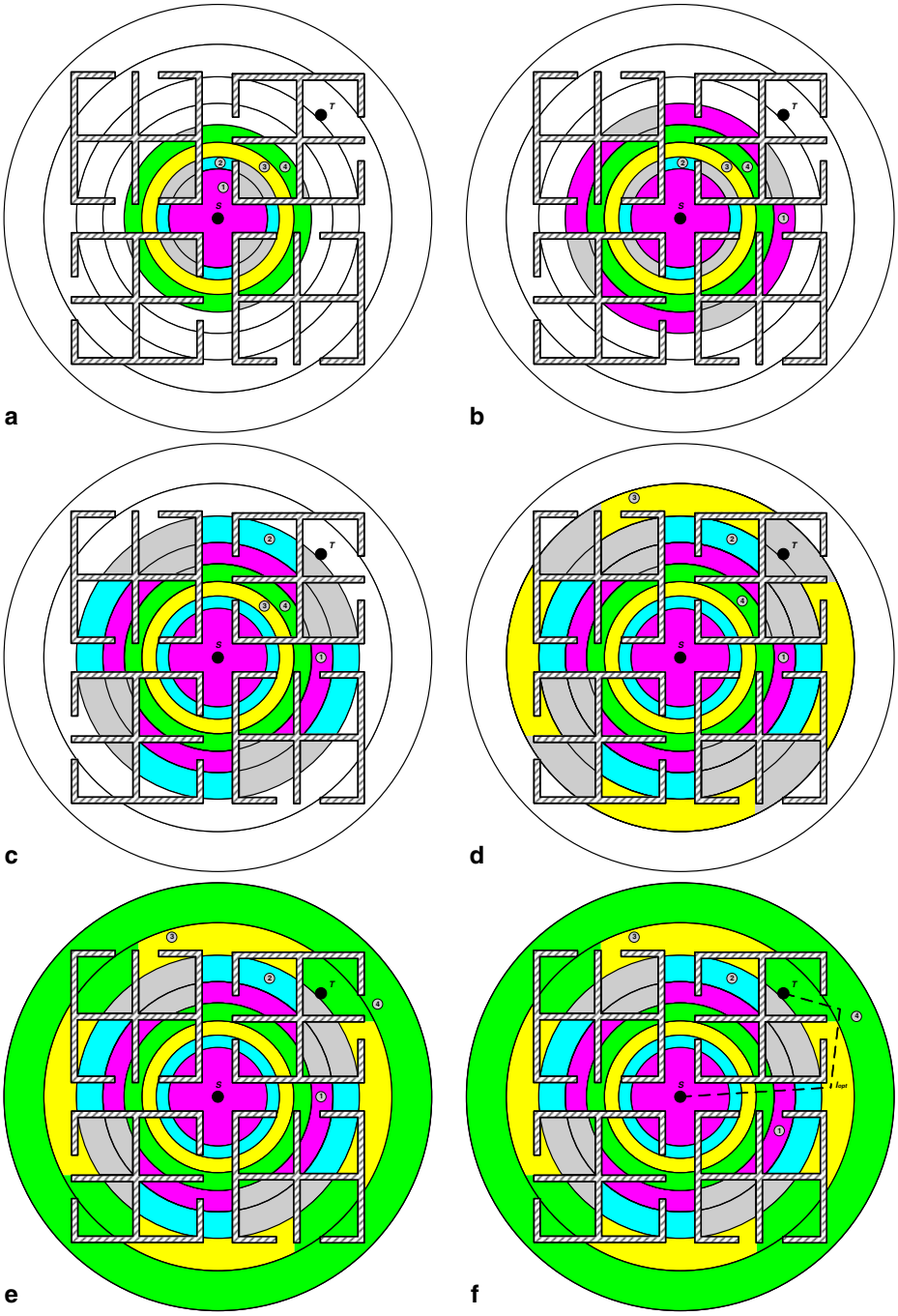

Fig. 6 **a–f** Six local steps of *MRSAM* simulation

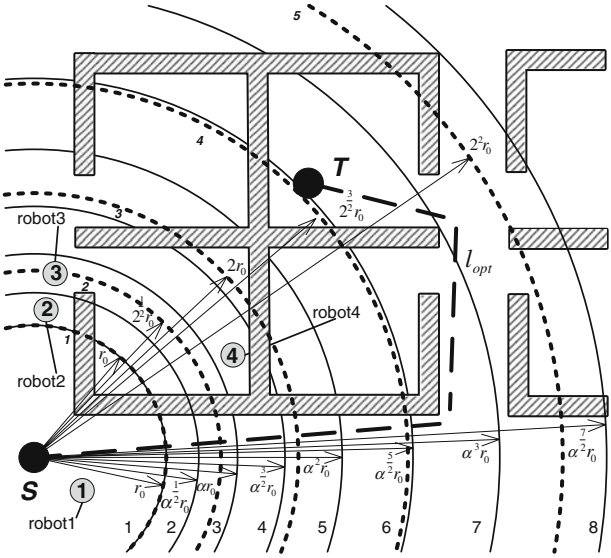

Fig. 7 The simulation environment, start and target points, the four robots initial position and two sets of search discs for two cases of *MRSAM* execution. The five *dashed circles* and the numbers attached to, denote the search discs for a single robot execution, and the eight *solid circles* denote the four-robot execution of *MRSAM*

5.3.5 Communication and centralization

Communication in *MRBUG* is needed only at termination, where all the robots must get the stop signal. The exchange of information between robots of the same pair in *PBUG1* is done only upon meeting, which means that the communication does not have to be wireless. However, communication between the robot pairs and wireless communication inside a pair will enhance the performance and the robustness, as discussed in the previous subsection. Since the robots are getting the necessary information at the beginning (the number of robot pairs for calculation of the multiplication factor,α, and the initial ellipses and directions assignment), and the fact that they execute *MRBUG* alone (in pairs) without needing any other information from the other robots or from any outer agent, *MRBUG* can be considered decentralized.

6 Simulations

6.1 *MRSAM* simulation setup

In the following simulation *MRSAM* algorithm launches 4 robots from a starting point S to search for the target T in an unknown office-cubical environment[8] as depicted in Fig. 7. The area multiplication factor for $n = 4$ is $\alpha = 1.495$, and $\sqrt{\alpha} = 1.223$, and in the initial global step $k = 1$ (local step $i = 1, 2, 3, 4$) each robot is assigned to one of the first four discs according to its number. In this early stage of the algorithm all the robots are assigned to a series of search discs until termination. It can be seen (although the robots do not know it yet) that the target resides in disc 7,

[8]This is merely a quarter of the symmetric environment.

which belongs to robot 3. The target is unreachable to robot 3 from disc 7 since the passage is not wide enough.

6.2 *MRSAM* simulation results

In the beginning of the simulation, each robot searches for the target until it covers all the reachable portion of the disc it is assigned to. The area that was not reachable in the current step, but is connected, like the gray areas depicted in Fig. 6a, will be covered in the next steps. Robot 1 finishes its local step in the first place and thus start its next global step, $k = 2$, which is the fifth local step, $i = 5$, searching in disc 5. Now the entire area of the first disc can be covered, yet some parts of disc 5 cannot be covered (Fig. 6b). In the next two steps, Robot 2 finishes its disc coverage and moves on to disc 6 ($k = 2, i = 6$) (Fig. 6c), and Robot 3 moves on to disc 7 ($k = 2$, $i = 7$) (Fig. 6d). In both steps, again, some parts cannot be reached and the target in particular. At last, Robot 4 reaches the next global step ($k = 2, i = 8$), where it moves to search in disc 8, now it can search in the previously unreachable areas that lie in disc 7 and there it reaches the target (Fig. 6e).

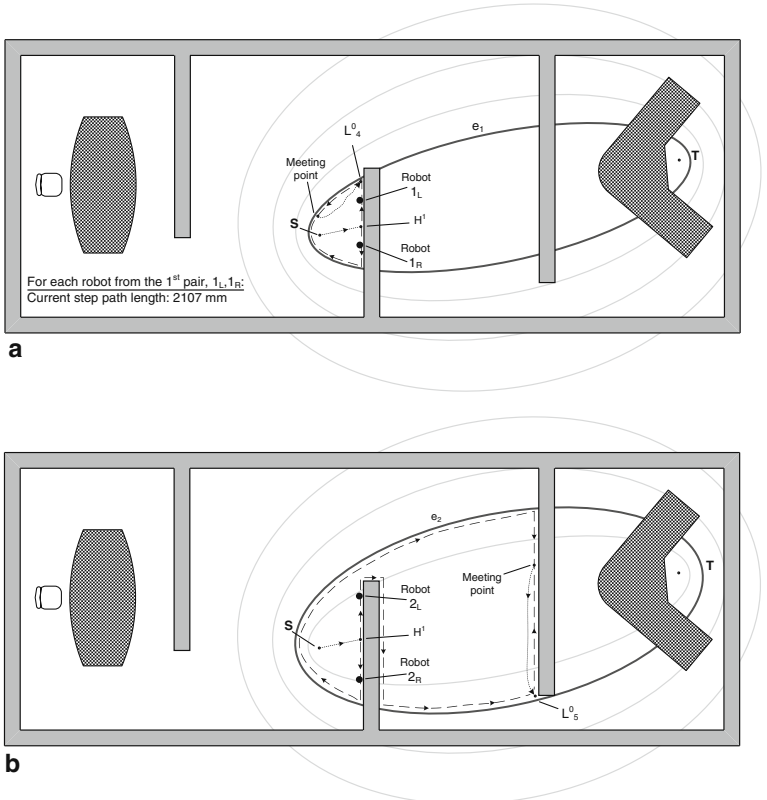

Fig. 8 **a**, **b** The first two steps of *MRBUG* simulation executing three pairs of robots. The *dense dashed lines* denote a mutual path of a robot pair

We simulated an execution of *MRSAM* with four robots, and compared it to execution with one robot on the same environment, including common starting and target points and identical initial disc. l_{opt} is marked in Fig. 6f and in Fig. 7 with bold dashed line. For the initial radius $r_0 = 22\,\text{cm}$ and $D = 5\,\text{cm}$ the simulation results are as follows. The optimal off-line solution is $l_{\text{opt}} = 126.8\,\text{cm}$, the path length generated by Robot 4, which found the target, during *MRSAM* execution is $l_j = 12121\,\text{cm} = 0.75\,l_{\text{opt}}^2\,\text{cm}$, and the path length when running with one robot is $l = 18730\,\text{cm} = 1.17\,l_{\text{opt}}^2\,\text{cm}$. These results show that *MRSAM* execution with 4 robots was 1.545 times faster than single robot execution. Observing Table 1, one may ask why *MRSAM* is not four times faster? The answer lies in the understanding of two factors, the specific environment's geometry, and the initial radius r_0 that was chosen. It should be clear that the relative performance values in Table 1 are calculated for executions in worst case scenarios. In real environments, the actual performance is better than in the worst case. A difference between the performance improvement of a single robot execution and of *MRSAM* execution might lead to variations such as seen above. The size of the initial r_0 affects the performance in two ways. First, consider the minimal disc which contains l_{opt}, and let r_{opt} denote its radius, since r_0 which is initialized to that value, implies that this disc will be the first and the only one

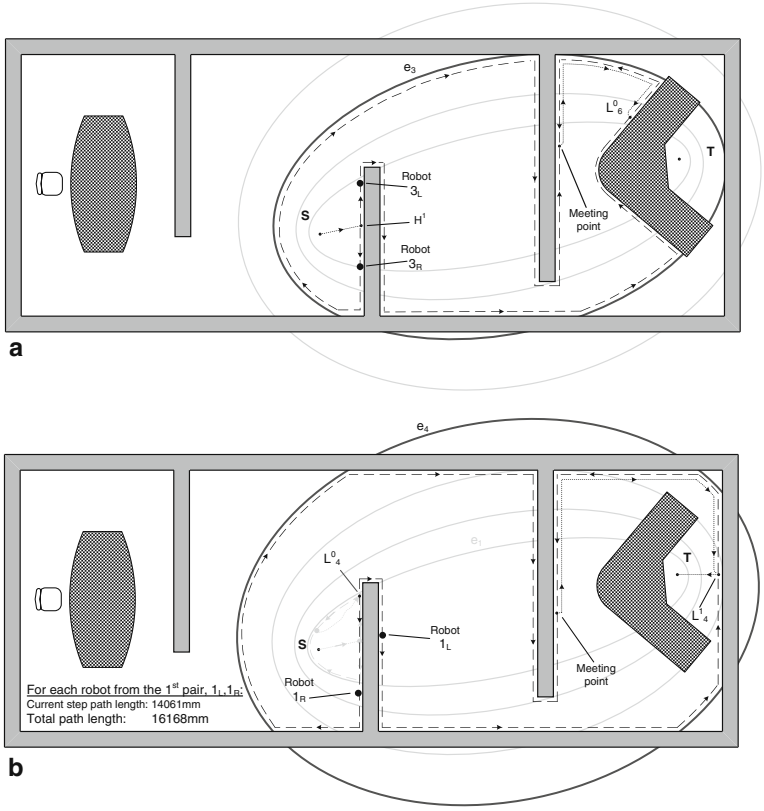

Fig. 9 **a**, **b** The last two steps of *MRBUG* simulation executing three pairs of robots

to be covered prior to finding the target. In cases where r_0 gets relatively big values up to r_{opt}, the actual performance can be reduced up to half of the upper bound, owing to the fact that the small terms do not appear in the sum of the geometric series of the areas covered by the robot which found the target.

6.3 *MRBUG* simulation setup

MRBUG simulation was executed in an office like environment consisting of four rooms with two desks. Three pairs of robots were launched from the same point in the left middle room with a mission to reach the target which lies behind a desk in the rightmost room. The start and the target points form the focal points of all the ellipses, and the distance between them equals $2c$ which defines the ellipse parameter c. This parameter is related to the ellipses semimajor and semiminor axes a and b by the following equation, $c = \sqrt{a^2 - b^2}$. Another important property of the ellipse is its area, which, for ellipse i equals $A(i) = \pi a_i b_i$. In *MRBUG* two more properties are added, the first is the area multiplication factor, $A(i) = \alpha A(i-1)$, for $i = 1, 2, \ldots$, and the second property is that c is constant for each problem and environment and in this simulation it equals $c = 1,753$ mm. The semiminor axis is defined to be $b_0 = 615$ mm and thus the ellipse is completely defined, a_0 and $A(0)$ are calculated from c

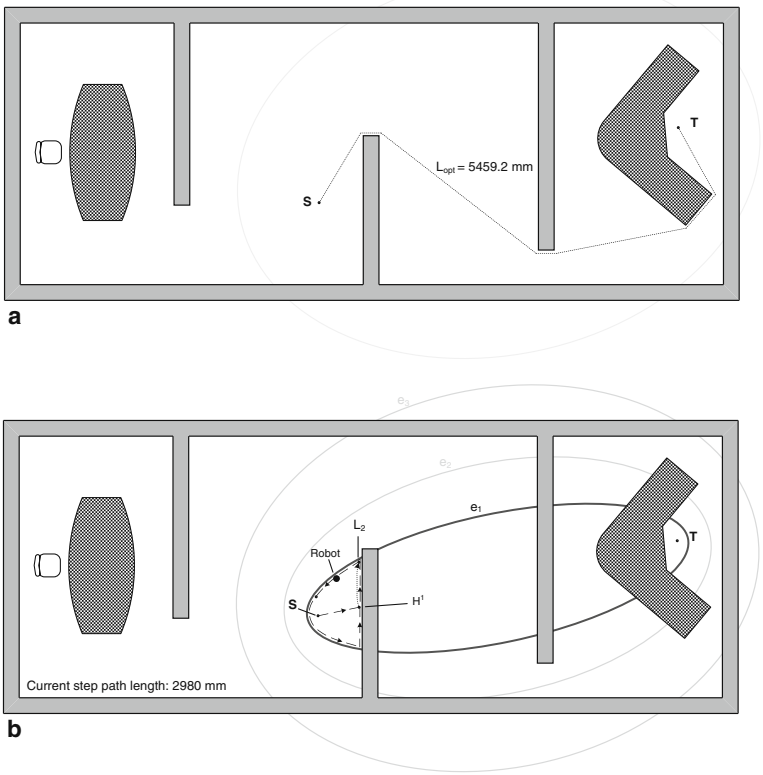

Fig. 10 The optimal off-line solution l_{opt} **a**, first step of *CBUG* simulation **b**

and b_i, $a_0 = 1,858$ mm and $A(0) = 30,560$ mm^2. After calculating α, $\alpha = 1.587$, the rest of the ellipses properties can be calculated.

6.4 *MRBUG* simulation results

The first step of *MRBUG* simulation for each pair of robots is depicted in Fig. 8. It can be observed that each robot pair cannot reach the target in its initial step, but, robot pair No. 3 became closer (Fig. 9a) to the target than robot pair No. 2 (Fig. 8b) which became closer to the target than the first robot pair (Fig. 8a).

The first robot pair to move to the next step and search for the path to target in the next ellipse is obviously robot pair No. 1 which continues the search in the fourth ellipse and starts this step from a closer point to the target, Fig. 9b. In this step robot pair No. 1 reach the target, and the path length of each of the robots equals $l = 16,168$ mm. The optimal off-line solution is depicted in Fig. 10a and $l_{\text{opt}} = 5,459$ mm. Thus, l can be represented as a function of l^2_{opt}, $l = 5.4 \cdot 10^{-4} l^2_{\text{opt}}$.

Comparing with the simulation of *CBUG* (Figs. 10b and 11) in the same environment as the *MRBUG* simulation, where *CBUG* uses only one robot and its area multiplication factor equals 2, one can see that it took *CBUG* two ellipse

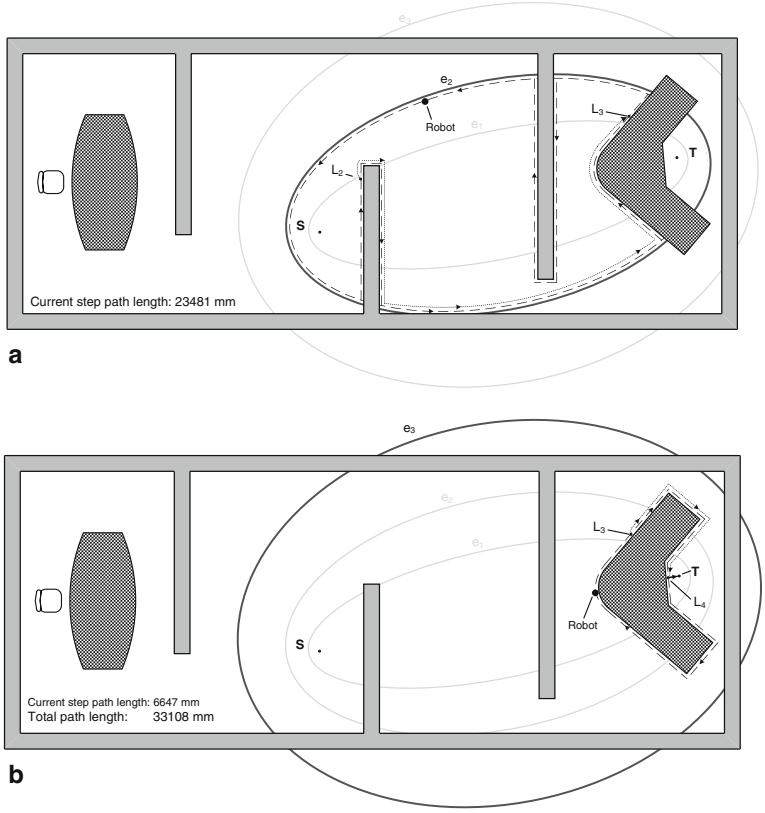

Fig. 11 **a**, **b** *CBUG* simulation in the same environment and configuration as in *MRBUG* simulation

multiplication prior to reaching the target and it produced a path length equals $l_{CBUG} = 33,108$, which is about 2 times longer than the path length produced by *MRBUG*. It should be noted, however, that as predicted, though the performance in *MRBUG* was doubled compared with *CBUG*, the number of robots in *MRBUG* was six times greater than in *CBUG*.

7 Conclusion

The notion of competitive complexity classes generalizes the traditional notion of linear competitiveness to a pair of bounds which up to constant coefficients satisfy the same functional relationship between on-line performance and off-line optimal solution. In particular, we have shown that on-line multi mobile robot navigation to a target belongs to the quadratic competitive complexity class. *MRSAM* and *MRBUG* algorithms achieves the optimal quadratic bound while requiring only a linear amount of memory.

The following are some related open problems for further research. First, *MRSAM* and *MRBUG* assumes tactile senors. More sophisticated sensors such as vision and laser sensors do not have a significant advantage on tactile sensors in highly congested environments. However, practical environments tend to be reasonably sparse, and an adaptation of *MRSAM* and *MRBUG* to such sensors is an important open problem. Second, the constant coefficients in the quadratic upper bound on *MRSAM* and in the quadratic universal lower bound differ by values of $\frac{3}{2}(n+1)^{\frac{n+1}{n}}$ and for *MRBUG* they differ by values of $\frac{3(1+\pi)^2}{2}(n+1)^{\frac{n+1}{n}}$. Closing of this gap is a major challenge that can yield new algorithms that possess the quadratic competitiveness of *MRSAM* but perform much better on average. Last, we assumed linear on-board memory. However, many mobile robot tasks are sufficiently complex as to allow only constant memory. Given this stricter memory limitation, one must re-explore the competitive complexity class of the basic problem considered in this paper.

Acknowledgements The authors would like to thank Yoav Gabriely and Elon Rimon for their helpful contribution to this article.

References

1. Antonelli, G., Arrichiello, F., Chiaverini, S., Setola, R.: A self-configuring MANET for coverage area adaptation through kinematic control of a platoon of mobile robot. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 1307–1312. Edmonton, Alberta, Canada (2005)
2. Arkin, R.C., Diaz, J.: Line-of-sight constrained exploration for reactive multiagent robotic teams. In: 7th International Workshop on Advanced Motion Control, pp. 455–461 (2002)
3. Baeza-Yates, R.A., Culberson, J.C., Rawlins, G.J.E.: Searching in the plane. Inf. Comput. **106**(2), 234–252 (1993)
4. Chen, J., Li, L.-R.: Path planning protocol for collaborative multi-robot systems. In: Proceedings 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 721–726. Espoo, Finland (2005)
5. Chung, T.H., Burdick, J.W., Murray, R.M.: A decentralized motion coordination strategy for dynamic target tracking. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2416– 2422 (2006)

6. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: Collaborative multi-robot localization. In: Proc. of the 23rd German Conference on Artificial Intelligence (KI), Germany (1999)
7. Franchi, A., Freda, L., Oriolo, G., Vendittelli, M.: A randomized strategy for cooperative robot exploration. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 768–774 (2007)
8. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Ann. Math. Artif. Intell. **31**, 77–98 (2001)
9. Gabriely, Y., Rimon, E.: Competitive on-line coverage of grid environments by a mobile robot. Comput. Geom. Theory Appl. **24**(3), 197–224 (2003)
10. Gabriely, Y., Rimon, E.: Competitive complexity of mobile robot on line motion planning problems. In: Workshop on Algorithmic Foundations of Robotics, pp. 249–264 (2004)
11. Gabriely, Y., Rimon, E.: CBUG: a quadratically competitive mobile robot navigation algorithm. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 954–960 (2005)
12. Ge, S.S., Fua, C.-H.: Complete multi-robot coverage of unknown environments with minimum repeated coverage. In: Proc. IEEE Int. Conf. on Robotics and Automation (2005)
13. Gonzalez, E., Alvarez, O., Diaz, Y., Parra, C., Bustacara, C.: BSA: a complete coverage algorithm. In: Proc. IEEE Int. Conf. on Robotics and Automation (2005)
14. Guo, Y., Parker, L.E.: A distributed and optimal motion planning approach for multiple mobile robots. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2612–2619. Washington, DC (2002)
15. Hazon, N., Kaminka, G.A.: Redundancy, efficiency, and robustness in multi-robot coverage. In: Proc. IEEE Int. Conf. on Robotics and Automation (2005)
16. Icking, C., Kamphans, T., Klein, R., Langetepe, E.: On the competitive complexity of navigation tasks. In: Revised Papers from the International Workshop on Sensor Based Intelligent Robots, pp. 245–258. Springer, London, UK (2002)
17. Icking, C., Klein, R.: Competitive strategies for autonomous systems. In: Bunke, H., Kanade, T., Noltemeier, H. (eds.) Modelling and Planning for Sensor Based Intelligent Robot Systems, pp. 23–40. World Scientific, Singapore (1995)
18. Koenig, S., Liu, Y.: Terrain coverage with ant robots: a simulation study. In: AGENTS'01: Proceedings of the Fifth International Conference on Autonomous Agents, pp. 600–607. ACM, New York, NY, USA (2001)
19. Kong, C.S., Peng, N.A., Rekleitis, I.: Distributed coverage with multi-robot system. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2423–2429 (2006)
20. Kurabayashi, D., Ota, J., Arai, T., Yoshida, E.: Cooperative sweeping by multiple mobile robots. In: Proc. IEEE Int. Conf. on Robotics and Automation. Minneapolis, Minnesota (1996)
21. Latimer IV, D.T., Srinivasa, S., Shue, V.L., Sonne, S., Choset, H., Hurst, A.: Towards sensor based coverage with robot teams. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation, vol. 1, pp. 961–967. IEEE (2002)
22. Lumelsky, V.J., Stepanov, A.A.: Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica **2**, 403–430 (1987)
23. Pinto, A.G., Fraisse, P., Zapata, R.: An decentralized adaptive trajectory planning approach for a group of mobile robots. In: Proceedings of Towards Autonomous Robotic Systems, pp. 65–72 (2005)
24. Rekleitis, I., Lee-Shue, V., New, A.P., Choset, H.: Limited communication, multi-robot team based coverage. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 3462–3468. New Orleans, LA (2004)
25. Rekleitis, I.M., New, A.P., Choset, H.: Distributed coverage of unknown/unstructured environments by mobile sensor networks. In: Schultz, A.C., Parker, L.E., Schneider, F. (eds.) 3rd International NRL Workshop on Multi-Robot Systems, pp. 145–155. Kluwer, Washington, D.C. (2005)
26. Sarid, S., Shapiro, A., Gabriely, Y.: MRSAM: a quadratically competitive multi robots navigation algorithm. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2699–2704. Orlando, FL (2006)
27. Sgorbissa, A., Arkin, R.C.: Local navigation strategies for a team of robots. Robotica **21**(05), 461–473 (2003)
28. Sheng, W., Yang, Q., Tan, J., Xi, N.: Distributed multi-robot coordination in area exploration. Robot. Auton. Syst. **54**(12), 945–955 (2006)
29. Stachniss, C., Mozos, O.M., Burgard, W.: Speeding-up multi-robot exploration by considering semantic place information. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1692–1697 (2006)
30. Williams, K., Burdick, J: Multi-robot boundary coverage with plan revision. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1716–1723 (2006)